ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

Thí nghiệm Hệ điều hành (CO2018)

Báo cáo bài tập thí nghiệm

Lab 1

Giảng viên: Nguyễn Quang Hùng, CSE-HCMUT

Sinh viên: Nguyễn Phúc Nhân - 2312438

HO CHI MINH CITY, MARCH 2025

Table of Contents

1 Bài tập 3.6

1.1 Tóm tắt yêu cầu

Viết một chương trình máy tính cơ bản bằng Shell Script đáp ứng được các yêu cầu sau:

- Dữ liệu nhập vào là một biểu thức, các phần tử trong biểu thức được phân cách bằng khoảng trắng.
- Hỗ trợ 5 phép tính: +, -, *, /, %. Lưu trữ kết quả gần nhất vào biến `ANS`, có thể dùng biến này cho những lần tính tiếp theo.
- Kết quả `ANS` được khởi tạo bằng **0** và không mất khi khởi động lại chương trình.
- Lưu trữ **5 phép tính thành công gần nhất** và khi dùng lệnh `HIST` sẽ xem lại được 5 lệnh gần nhất.
- Khi đợi người dùng nhập phép tính, sẽ hiển thị » .
- Sau khi nhập phép tính hợp lệ, người dùng nhấn **ENTER** để hiển thị kết quả phép tính, kết quả in ở dòng dưới.

Các yêu cầu về hiển thị:

1. Các số không phải số nguyên thì làm tròn kết quả đến **2 chữ số thập phân**.
2. Sau khi tính toán xong, nhấn **ENTER** để tiếp tục bước tính tiếp theo. Màn hình sẽ bị xoá.
3. Với những phép tính không hợp lệ (chia cho 0) thì in ra **MATH ERROR**.
4. Với những đầu vào không hợp lệ, hiển thị **SYNTAX ERROR**.
5. Lệnh `EXIT` để thoát chương trình.

1.2 Hiện thực

1.2.1 Nhận dữ liệu từ người dùng

Sử dụng lệnh `read` để nhận dữ liệu:

```
# GET USER INPUT
read -p ">> " exp_str
```

1.2.2 Kiểm tra và tính toán biểu thức

Sử dụng `bc` và `awk` để xử lý các phép tính.

1.2.3 Lưu kết quả gần nhất vào biến `ANS`

Kết quả cuối cùng được lưu vào tệp `cache.out` để không bị mất khi khởi động lại.

1.2.4 Xóa màn hình sau mỗi phép tính

Dùng `clear` để xoá màn hình trước khi chờ nhập phép tính mới.

1.2.5 Lưu trữ và hiển thị lịch sử 5 phép tính gần nhất

Khai báo mảng `HISTORY` và sử dụng lệnh `HIST` để hiển thị lịch sử.

1.2.6 Xử lý lệnh `EXIT` và `HIST`

Dùng `case` để kiểm tra và xử lý hai lệnh trên.

1.2.7 Mã hoàn chỉnh

Đoạn code hoàn chỉnh của chương trình

```
#!/bin/bash
declare -a HISTORY
LAST_ANS_FILE="cache.out"
LAST_ANS="0"

if [ -f "$LAST_ANS_FILE" ]; then
    LAST_ANS=$(cat "$LAST_ANS_FILE")
else
    LAST_ANS="0"
fi

while true; do
    # * USER INPUT
    read -p ">> " exp_str

    # * Process special commands
    case "$exp_str" in
        EXIT)
            exit 0
            ;;
        HIST)
            if [ ${#HISTORY[@]} -gt 5 ]; then
                for ((i=${#HISTORY[@]}-5; i<${#HISTORY[@]}; i++)); do
                    echo "${HISTORY[$i]}"
                done
            else
                for line in "${HISTORY[@]}"; do
                    echo "$line"
                done
            fi
            read -s
            clear
            continue
            ;;
    esac

    # Store input
    expr=$(echo "$exp_str" | sed "s/ANS/$LAST_ANS/g")

    # Validate equation
    if echo "$expr" | grep -q "[^0-9+*/%() -]"; then
        echo "SYNTAX ERROR"
        exit 1
    fi

    # Check if the operation contains '%'
    if echo "$expr" | grep -q "%"; then
        result=$(awk "BEGIN { print $expr }")
        echo "" > err.tmp
    else
        result=$(echo "scale=2; $expr" | bc 2>err.tmp | sed 's/\.00$//' | sed 's/^\.0*/0./')
    fi

    err=$(cat err.tmp)
```

```
    # Check for errors
    if [[ -n "$err" ]]; then
        if echo "$err" | grep -qi "division by zero"; then
            echo "MATH ERROR"
        else
            echo "SYNTAX ERROR"
        fi
        read -s

        clear
        continue
    fi

    # Print the result
    echo "$result"

    # Save the last answer
    LAST_ANS="$result"
    echo "$LAST_ANS" > "$LAST_ANS_FILE"
    HISTORY+=("$exp_str = $result")

    read -s

    clear
done
```

2 Bài tập 5.3

2.1 Tóm tắt yêu cầu

Xây dựng lại một chương trình máy tính như bài trước nhưng không có lệnh `HIST` bằng ngôn ngữ C với các yêu cầu sau:

- Tạo một Makefile với ít nhất 2 target: `all` và `clean`.
- Tên file thực thi là `calc`.
- Hàm `main()` được cài đặt trong file `calc.c`, trong khi các hàm xử lý tính toán được tách ra các file nguồn khác.
- Các yêu cầu nhập, xuất, xử lý biểu thức, thay thế `ANS` và kiểm tra lỗi (như `SYNTAX ERROR` và `MATH ERROR`) giống như phiên bản Shell Script.

2.2 Hiện thực

2.2.1 Tạo Makefile

Tạo file `Makefile` để biên dịch chương trình với các target `all` và `clean`.

```
CC = gcc
CFLAGS = -O2 -g

OBJS = calc.o calc_logic.o

all: calc

calc: $(OBJS)
 $(CC) $(CFLAGS) -o calc $(OBJS)

calc.o: calc.c calc_logic.h
```

```
 $(CC) $(CFLAGS) -c calc.c

calc_logic.o: calc_logic.c calc_logic.h
 $(CC) $(CFLAGS) -c calc_logic.c

clean:
 rm -f calc $(OBJS)
```

Target `all` sẽ tạo ra file thực thi `calc` còn target `clean` dùng để dọn dẹp các file đối tượng.

2.2.2 Cấu trúc chương trình

Chương trình được chia thành các file sau:

- `calc.c`: Chứa hàm `main()`, xử lý đầu vào từ người dùng và điều khiển luồng chương trình.
- `calc_logic.c`: Chứa các hàm thực hiện các phép tính (cộng, trừ, nhân, chia, lấy phần dư).
- `calc_logic.h`: Chứa các khai báo hàm và định nghĩa dùng chung giữa các file nguồn.

2.2.3 Xử lý đầu vào và tính toán

- **Nhận dữ liệu đầu vào:** Chương trình hiển thị "`» `" để đợi người dùng nhập biểu thức. Các thành phần của biểu thức được phân cách bằng khoảng trắng.
- **Xử lý biểu thức:**
  - Nếu biểu thức chứa từ khóa `ANS` thì sẽ được thay thế bằng giá trị của biến `ANS` (được lưu trong file `cache.out`, nếu file không tồn tại thì `ANS` khởi tạo bằng 0).
  - Chương trình kiểm tra tính hợp lệ của biểu thức: chỉ cho phép các ký tự số, dấu cách, dấu ngoặc và các toán tử `+`, `-`, `*`, `/`, `%` và dấu chấm.
  - Nếu gặp lỗi nhập liệu, chương trình in ra `SYNTAX ERROR`.
  - Nếu gặp lỗi trong quá trình tính (như chia cho 0) thì in ra `MATH ERROR`.
  - Các phép tính với số không phải số nguyên được làm tròn đến 2 chữ số thập phân.
- **Xuất kết quả:** Kết quả của phép tính được in ra ngay dưới dòng biểu thức đã nhập.

2.2.4 Xử lý lệnh đặc biệt

- Lệnh `EXIT` sẽ thoát khỏi chương trình.
- Chức năng `HIST` không được triển khai trong phiên bản C này.

2.3 Kết quả thử nghiệm

Sau khi biên dịch bằng Makefile (sử dụng lệnh `make`), chương trình chạy được và cho kết quả như sau:

```
>> 5 + 3
8

>> 15 / 4
3.75

>> ANS * 2
7.50

>> 5 / 0
MATH ERROR

>> 6 @ 2
SYNTAX ERROR

>> EXIT
```

Như vậy, chương trình:

- Hiển thị "» " chờ nhập liệu.
- Thực hiện đúng các phép tính với kết quả được làm tròn đến 2 chữ số thập phân.
- Xử lý chính xác các lỗi nhập liệu và lỗi toán học.
- Lưu trữ và sử dụng giá trị của `ANS` cho các lần tính tiếp theo.